IN THE CLAIMS:

Please and amend the claims as follows:

1. (Currently Amended) A method of determining variables to update in a debugging environment, the method comprising executing at least one of a first task when a run command is received and a second task when a set step command for a statement is received, wherein:

(a) the first task comprises determining a first kill variables set comprising only those variables being monitored by the debugging environment whose values may ~~which may be affected by~~ become out of sync with the corresponding values of those monitored variables while being used by a program being debugged during the execution of the program from a particular point of the program to a breakpoint, wherein the breakpoint can be encountered during execution of the program from the particular point; and

(b) the second task comprises determining a second kill variables set comprising only those variables being monitored by the debugging environment whose values may become out of sync with the corresponding values of those monitored variables while being used by a program being debugged ~~which may be affected~~ by ~~execution~~ executing of the statement.

2. (Original) The method of claim 1, wherein determining the first kill variables set comprises referencing a kill set of a statement, wherein the kill set of the statement contains variables affected by the statement.

3. (Original) The method of claim 1, further comprising, prior to determining the first kill set, determining whether any breakpoints may be encountered during subsequent continuing execution from the particular point.

4. (Original) The method of claim 1, further comprising, where the second task is executed:

setting a step operation for the statement;

executing the step operation; and

updating on a user interface only the variables contained in the second kill variables set.

5. (Original) The method of claim 1, wherein the first task is performed for a plurality of breakpoints, each of which may be encountered during execution of the program from the particular point in the program, whereby an instance of the first kill variables set is provided for each of the plurality of breakpoints.

6. (Original) The method of claim 5, further comprising, where the first task is executed:

executing the run command;

hitting a particular breakpoint of the plurality of breakpoints; and

updating on a user interface only the variables contained in a respective first kill variables set associated with the particular breakpoint.

7. (Original) The method of claim 1, further comprising, where the first task is executed:

executing the run command;

hitting the breakpoint; and

updating on a user interface only the variables contained in the first kill variables set.

8. (Original) The method of claim 7, wherein the updating is only performed if the first task has completed execution.

9. (Currently Amended) The method of claim 1, wherein determining the first kill variables set comprises:

if a node of a control flow graph containing the particular point of the program contains at least one breakpoint, determining kill variables of each statement of the

node from the particular point up to a first encountered breakpoint, wherein the first encountered breakpoint is a first breakpoint encountered when traversing the node from the particular point and wherein the kill variables of a respective statement are those variables which may be affected by execution of the respective statement; and then

generating the first kill variables set by associating all determined local kill variables with the first encountered breakpoint.

10. (Original) The method of claim 9, further comprising updating on a user interface only the variables contained in the first kill variables set.

11. (Currently Amended) The method of claim 9, wherein determining the first kill variables set further comprises, if the node does not contain at least one other breakpoint:

beginning with a root node, marking each node of [[a]] the control flow graph which may be exited and reentered during execution of the program from the particular point;

for each marked node, marking all variables of the control flow graph as a kill variable; and

beginning with the node containing the particular point of the program, generating a list of unmarked nodes which may be reached during execution from the particular point.

12. (Original) The method of claim 11, wherein generating the list comprises:

traversing the control flow graph from the node containing the particular point of the program to each subsequent node by following program control flow defined by arcs;

adding each encountered node to the list if the encountered node is not marked and is not already in the list;

determining whether the encountered node contains a breakpoint; and

if so, terminating a traversal along a current traversal path.

13. (Original) The method of claim 11, further comprising:

propagating, to each unmarked node in the list, a propagated kill set from a preceding unmarked node in the control flow graph, wherein the propagated kill set contains a sum of all kill variables associated with all statements of the preceding unmarked node and all propagated kill variables propagated to the preceding unmarked node.

14. (Currently Amended) A computer readable medium containing a program which, when executed by a processor, performs operations for determining variables to update in a debugging environment, the operations comprising executing at least one of a first task when a run command is received and a second task when a set step command for a statement is received, wherein:

(a) the first task comprises determining a first kill variables set comprising only those variables being monitored by the debugging environment whose values may ~~which may be affected by~~ become out of sync with the corresponding values of those monitored variables while being used by a program being debugged during the execution of the program from a particular point of the program to a breakpoint, wherein the breakpoint can be encountered during execution of the program from the particular point; and

(b) the second task comprises determining a second kill variables set comprising only those variables being monitored by the debugging environment whose values may become out of sync with the corresponding values of those monitored variables while being used by a program being debugged ~~which may be affected~~ by ~~execution~~ executing of the statement.

15. (Original) The computer readable medium of claim 14, wherein determining the first kill variables set comprises referencing a kill set of a statement, wherein the kill set of the statement contains variables affected by the statement.

16. (Original) The computer readable medium of claim 14, further comprising, prior to determining the first kill set, determining whether any breakpoints may be encountered during subsequent continuing execution from the particular point.

17. (Original) The computer readable medium of claim 14, further comprising, when the second task is executed:

setting a step operation for the statement;

executing the step operation; and

updating on a user interface only the variables contained in the second kill variables set.

18. (Original) The computer readable medium of claim 14, wherein the first task is performed for a plurality of breakpoints, each of which may be encountered during execution of the program from the particular point in the program, whereby an instance of the first kill variables set is provided for each of the plurality of breakpoints.

19. (Original) The computer readable medium of claim 18, further comprising, when the first task is executed:

executing the run command;

hitting a particular breakpoint of the plurality of breakpoints; and

updating on a user interface only the variables contained in a respective first kill variables set associated with the particular breakpoint.

20. (Original) The computer readable medium of claim 14, further comprising, where the first task is executed:

executing the run command;

hitting the breakpoint; and

updating on a user interface only the variables contained in the first kill variables set.

21. (Original) The computer readable medium of claim 20, wherein the updating is only performed if the first task has completed execution.

22. (Currently Amended) The computer readable medium of claim 14, wherein determining the first kill variables set comprises:

if a node of a control flow graph containing the particular point of the program contains at least one breakpoint, determining kill variables of each statement of the node from the particular point up to a first encountered breakpoint, wherein the first encountered breakpoint is a first breakpoint encountered when traversing the node from the particular point and wherein the kill variables of a respective statement are those variables which may be affected by execution of the respective statement; and then

generating the first kill variables set by associating all determined local kill variables with the first encountered breakpoint.

23. (Original) The computer readable medium of claim 22, further comprising updating on a user interface only the variables contained in the first kill variables set.

24. (Currently Amended) The computer readable medium of claim 22, wherein determining the first kill variables set further comprises, if the node does not contain at least one other breakpoint:

beginning with a root node, marking each node of [[a]] the control flow graph which may be exited and reentered during execution of the program from the particular point;

for each marked node, marking all variables of the control flow graph as a kill variable; and

beginning with the node containing the particular point of the program, generating a list of unmarked nodes which may be reached during execution from the particular point.

25. (Original) The computer readable medium of claim 24, wherein generating the list comprises:

traversing the control flow graph from the node containing the particular point of the program to each subsequent node by following program control flow defined by arcs;

adding each encountered node to the list if the encountered node is not marked and is not already in the list;

determining whether the encountered node contains a breakpoint; and

if so, terminating a traversal along a current traversal path.

26. (Original) The computer readable medium of claim 24, further comprising:

propagating, to each unmarked node in the list, a propagated kill set from a preceding unmarked node in the control flow graph, wherein the propagated kill set contains a sum of all kill variables associated with all statements of the preceding unmarked node and all propagated kill variables propagated to the preceding unmarked node.

27. (Currently Amended) A computer readable medium containing a program which, when executed by a processor, performs operations for determining variables to update in a debugging environment, the operations comprising:

executing a task when a run command is received, wherein the task comprises determining a kill variables set comprising only those variables being monitored by the debugging environment whose values may ~~which may be affected by~~ become out of sync with the corresponding values of those monitored variables while being used by a program being debugged during the execution of the program from, and including, a particular statement of the program to a breakpoint that can be encountered during execution of the program from the particular point; and

if the task has completed execution when the breakpoint is encountered during execution of the program, updating on a user interface only the variables contained in the kill variables set.[[.]]

28. (Original) The computer readable medium of claim 27, wherein determining the kill variables set comprises:

beginning with a root node, marking each node of a control flow graph which may be exited and reentered during execution of the program from the particular statement;

for each marked node, marking all variables of the control flow graph as a kill variable; and

beginning with the node containing the particular statement, generating a list of unmarked nodes which may be reached during execution from the particular statement.

29. (Original) The computer readable medium of claim 28, propagating, to each unmarked node in the list, a propagated kill set from a preceding unmarked node in the control flow graph, wherein the propagated kill set contains a sum of all kill variables associated with all statements of the preceding unmarked node and all propagated kill variables propagated to the preceding unmarked node.